Capitulo 1a

Licenciatura en Ciencias Computacionales Materia: Autómatas y Compiladores Profesor: Eduardo Cornejo Velázquez Alumno: Jorge Luis Ortega Pérez 6°3

Procesadores de lenguaje

Se da a entender que un compilador es u programa que puede leer un programa en un lenguaje (el lenguaje fuente) y traducirlo en un programa equivalente en otro lenguaje (e quaie destino):

Una de las funciones más importantes de compilador es reportar cualquier error en el programa fuente que detecte durante el roceso de traducción

Si el programa destino resulta ser ui programa ejecutable en lenguaje máquina, entonces se puede ejecutar para procesar las ntradas v producir salidas (resultados).

En vez de producir un programa destin como una traducción, el intérprete da la apariencia de ejecutar directamente las operaciones especificadas en el programa de origen (fuente) con las entradas oporcionadas por el usuario.

procesa mediante un

programa llamado

ensamblador, el cual

relocalizable como

El cargador reúne

todos los archivos

objeto ejecutables en

la memoria para su

ecución

código

produce

máguina

ı salida.

ene la tarea d programa de origen se confía algunas veces a un programa separado, llamado eprocesador.

l enlazador resuely las direcciones de memoria externas, en donde el código en un archivo puede hacer referencia a una ubicación en otro hivo

Estructura de un compilador

simple que mapea un programa fuente a un programa destino con equivalencia semántica, pero en ella existen dos ocesos: análisis y síntesis.

En el análisis se divide el programa fuente en componentes e impone una estructura gramatical sobre ellas, se le llama omúnmente el front-end del compilador;

n la síntesis se construye el progran destino deseado a partir de la representación intermedia y de la información en la tabla de símbolos, se le

a tarea de esta optimización es pod realizar transformaciones sobre representación intermedia, para que el backend pueda producir un mejor programa destino de lo que hubiera producido con una resentación intermedia sin optimizar

> as fases siguientes d compilador utilizan la estructura gramatical para ayudar a analizar el programa fuente y generar

programa destino.

Análisis de léxico

ste lee el fluio de caracteres qu componen el programa fuente y los agrupa en secuencias significativas, conocidas como lexemas. Para cada una, el analizador léxico produce como alida un token de la forma:

{nombre-token, valor-atributo}

n el token, el primer component

nombre-token es un símbolo abstracto

que se utiliza durante el análisis

sintáctico, y el segundo componente

valor-atributo apunta a una entrada en la

tabla de símbolos para este token. La

información de la entrada en la tabla de

símbolos se necesita para el análisis

mántico y la generación de código.

Análisis sintáctico

parser (analizador sintáctico) utiliza los

primeros componentes de los tokens

producidos por el analizador de léxico para

crear una representación intermedia en

forma de árbol que describa la estructura

Una representación típica es el árbo

sintáctico, en el cual cada nodo interior

representa una operación y los hijos del nodo

epresentan los argumentos de la operación.

amatical del flujo de tokens.

En está, la segunda fase del compilado

In algoritmo simple de generación d código intermedio, seguido de la optimización de código, es una manera razonable de obtener un buen código de

suma menos poder

lormalmente se dedica mucho tiempo e esta fase, debido a que se realizar optimizaciones que considerablemente el tiempo de ejecución sin que afecte demasiado la velocidad de la

Optimización

de código

se trata de mejorar el código intermedio, de

manera que se produzca un mejor código

destino. Por lo general, significa más rápido,

pero pueden lograrse más objetivos, como

in código más corto, o de destino que

Generación de código

El generador de código recibe como entrada una representación intermedia del programa uente y la asigna al lenguaje destino

el lenguaje destino es código máquin se seleccionan registros o ubicaciones (localidades) de memoria para cada una de las variables que utiliza el programa. Después, las instrucciones intermedias se traducen en secuencias de instrucciones máquina que realizan la misma tarea

Administración de la tabla de símbolos

que contiene un registro para cada nombre de variable, con campos para los atributos del nombre. La estructura de datos debe diseñarse de tal forma que permita al compilador buscar el registro para cada nombre, y almacenar ener datos de ese registro con rapidez.

Evolución de los lenguajes de programación

aparecieron en la década de 1940 y se programaban en lenguaje máquina, mediante secuencias de 0's y 1's que indicaban de manera explícita a la computadora las operaciones que oía ejecutar, y en qué orden.

Primera generación: Lenguaies de máquin

generación:

nivel como Fortran, Cobol, Lisp, C, C++, C#,

especializados para aplicaciones, como

Quinta generación: Lenguajes basados en

generación:

Cuarta

SOL v Postscript.

lógica, como Prolog

na función esencial de un compilador e registrar los nombres de las variables que se Segunda utilizan en el programa fuente, y recolectar información sobre varios atributos de cada • Tercera generación: Lenguajes de alto

Agrupamiento de fases en pasadas

de un compilador. En una implementación las actividades de varias fases pueder agruparse en una pasada, la cual lee un archivo de entrada y escribe en un archivo

jemplo: Las fases correspondientes front-end del análisis léxico, análisis sintáctico, análisis semántico y generación de código intermedio podrían agruparse una sola pasada.

front-end (para un lenguaje específico) back-end (para una máquina destino).

Impactos en el compilador

os lenguajes de programación y los compiladore están íntimamente relacionados; los avances en los lenguajes de programación impusieron nuevas demandas sobre los escritores de compiladores. Éstos tenían que idear algoritmos y representaciones para traducir y dar soporte a las nuevas características

esde la década de 1940, la arquitectura computadoras ha evolucionado también. Los escritores de compiladores no sólo tuvieron que rastrear las nuevas características de un lenguaje, sino que también tuvieron que idear algoritmos de traducción para aprovechar al máximo las nuevas aracterísticas del hardware.

Los compiladores permiten el uso eficiente de lenguajes de alto nivel al minimizar la sobrecarga de ejecución, y son cruciales para aprovechar las rquitecturas computacionales de alto rendimiento.

Un compilador debe traducir en forma correcta conjunto potencialmente infinito de programas que odrían escribirse en el lenguaje fuente.

Análisis semántico

la información en la tabla de símbolos para comprobar la consistencia semántica del programa ente con la definición del lenguaje.

También recopila información sobre el tipo y la guarda, ya sea en el árbol sintáctico o en la tabla de símbolos, para usarla más tarde durante la generación de código intermedio.

. Una parte importante del análisis semántico es la comprobación (verificación) de tipos. en donde el compilador verifica que cada perador tenga operandos que coincidan

> La especificación del lenguaje puede permitir ciertas conversiones de tipo nocidas como coerciones.

Generación de código intermedio

En el proceso de traducción de un programa, un compilador puede generar representaciones intermedias, como los árboles sintácticos, que se usan durante los análisis sintáctico y semántico.

Un código de tres direcciones consiste en una secuencia de instrucciones similares a ensamblador, con tres operandos por instrucción, cada uno puede actuar como un registro.

En primer lugar, cada instrucción de asignación de tres direcciones tiene, por lo menos, un operador del lado derecho, estas instrucciones corrigen el orden en el que se van a realizar las operaciones.

En segundo lugar, el compilador debe generar un nombre temporal para guardar el valor calculado por una instrucción de tres direcciones.

En tercer lugar, algunas "instrucciones de tres direcciones" como la primera y la última en la secuencia, tienen menos de tres operandos.

Herramientas de construcción de compiladores

desarrolladores de compiladore pueden aprovechar herramientas moderna como editores de lenguaje, depuradores, administradores de versiones, profilers y ornos de prueba seguros.

gunas herramientas de construcción d compiladores de uso común son:

- Generadores de
- sintácticos (parsers) Generadores de escáneres
- Motores de traducción orientados a la · Generadores de generadores de
- · Motores de análisis de flujos de datos

códiao

· Kits (conjuntos) de herramientas para la construcción de compiladores

t2 = id3 * t1

t3 = id2 + t2